

# Tema 1-1.

# Visualización Científica

Métodos Numéricos para la Computación

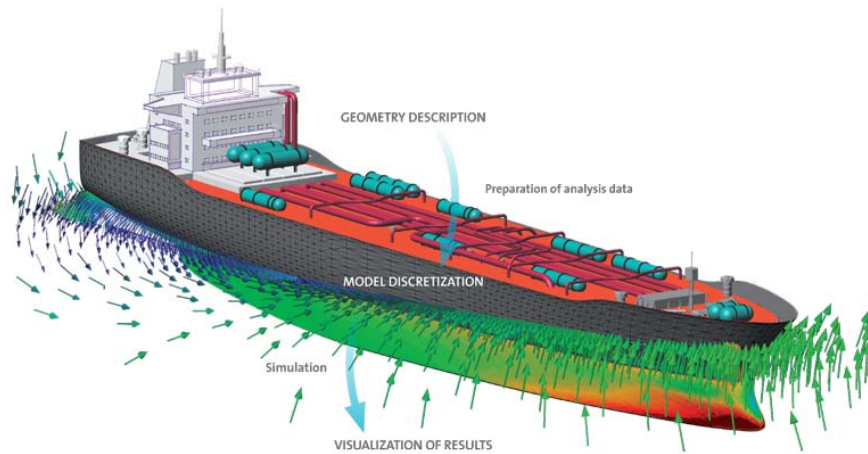
Grado en Ingeniería Informática  
Escuela de Ingeniería Informática  
Universidad de Las Palmas de Gran Canaria

Curso 2015/2016

# Índice del Tema

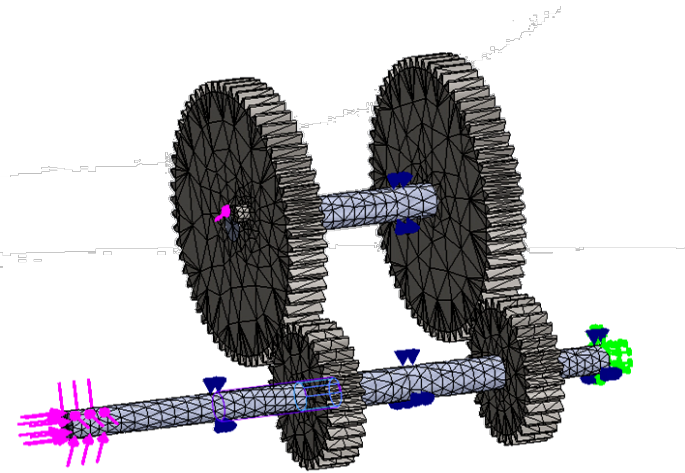
- **Tema 1-1.**
- Ciclo de procesos en Computación Científica
- Visualización Científica
- Visualización Científica en MATLAB

# Ciclo de Computación Científica

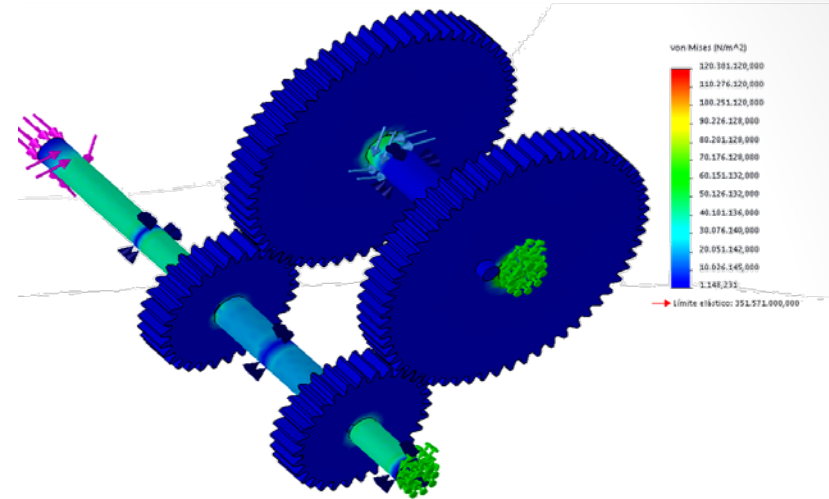


- Pre-procesado (Definición de la geometría, mallado/meshing)
- Simulación (Resolución de los modelos numéricos)
- Post-procesado (Visualización de los resultados)

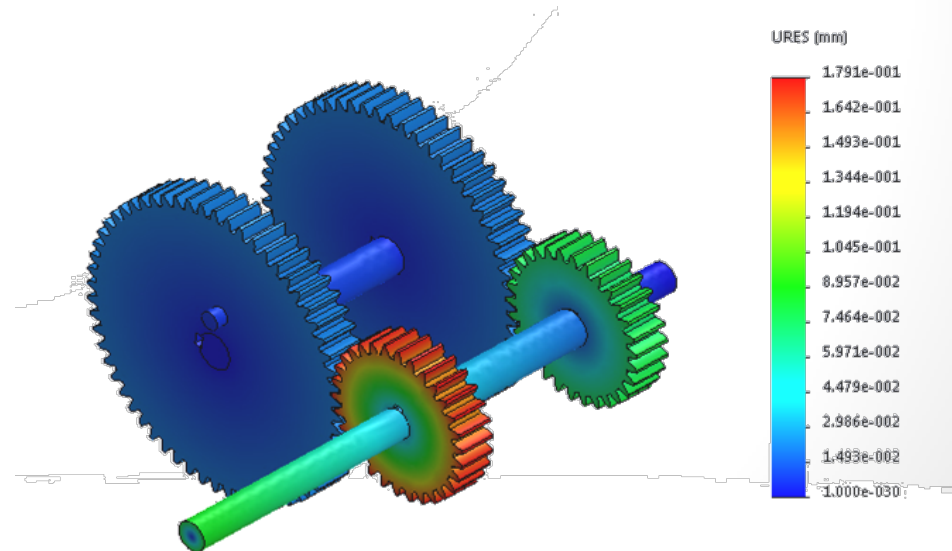
# Ejemplo de Diseño y Simulación



Diseño de un sistema mecánico incluyendo los esfuerzos y el mallado

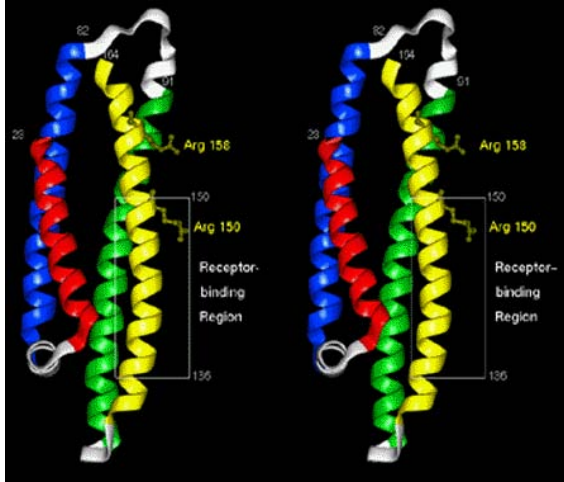


Visualización de datos de las tensiones

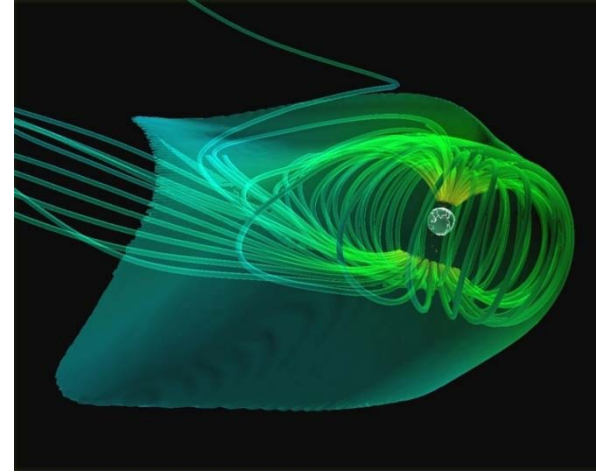


Visualización de datos de las deformaciones

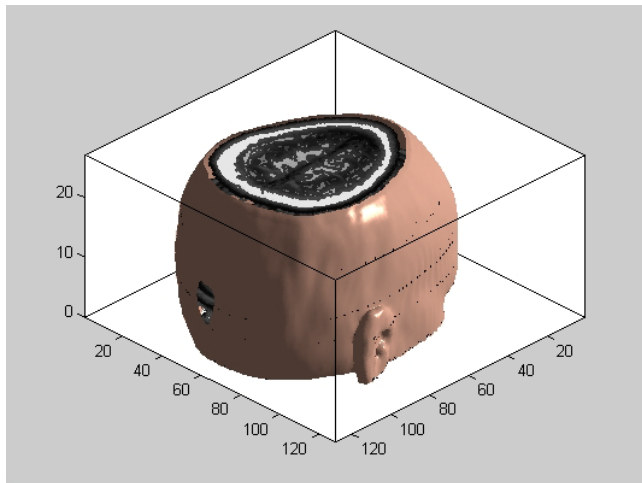
# Visualización Científica?



Visualización de la estructura 3D de una proteína



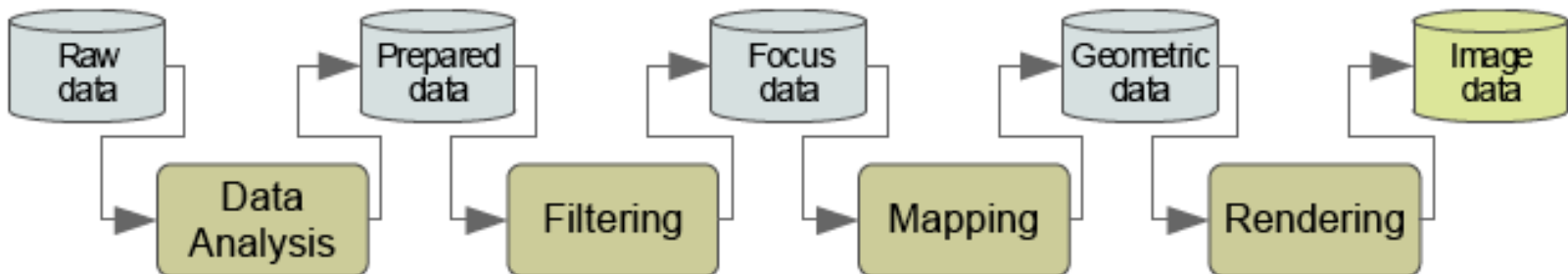
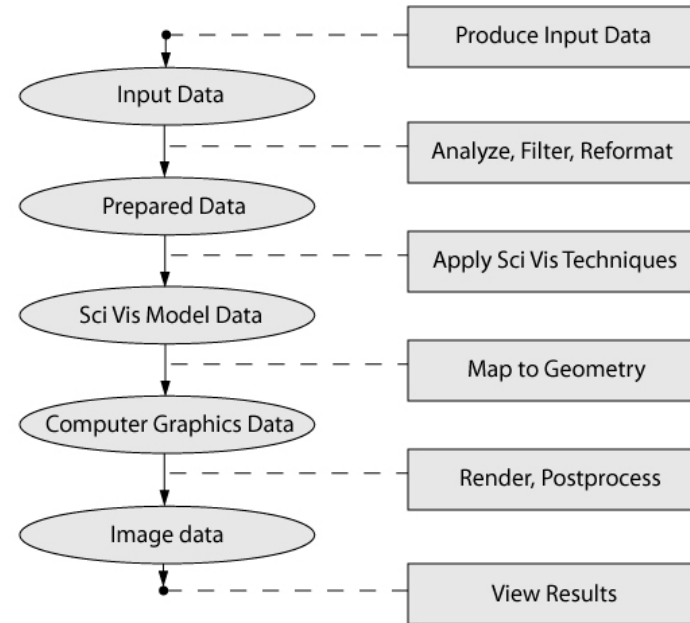
Visualización de las líneas de campo de la magnetosfera de la Tierra



Integración de imagen TAC y gráfico 3D

# Procesos en Visualización Científica

Una aplicación muy completa puede requerir diversas etapas que configuran un Pipeline o cadena de procesos

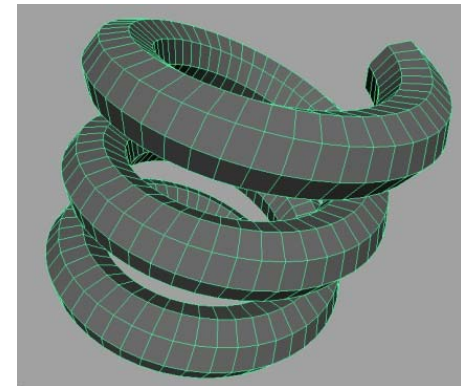


# Escalas de Presentación de datos

Datos matemáticos en presentación plana. Fáciles de interpretar por el investigador/profesional, pero de escaso valor para terceros.

```
191 214 206 113 134 135 101 120 135 102 119 147 113 129 165
117 131 170 125 138 170 158 168 195 219 227 246 244 251 255
249 254 255 197 202 222 157 163 187 156 161 190 165 172 200
218 224 246 250 255 255 252 255 251 251 255 252 245 249 250
251 255 255 188 192 204 196 200 212 234 238 250 177 181 192
111 114 133 122 121 155 154 151 194 182 176 220 243 240 255
253 252 255 250 251 246 254 254 252 255 255 255 255 255 255
254 254 254 255 255 255 255 255 255 254 254 254 255 255 255
249 249 249 249 249 249 252 252 252 255 221 155 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 253 253 253 252 252 252
252 252 252 253 253 253 255 255 255 255 255 255 255 255 255
```

Visualización científica. Focalizados en una interpretación cualitativa sencilla para el público objetivo de la ilustración. No requiere gran calidad visual.

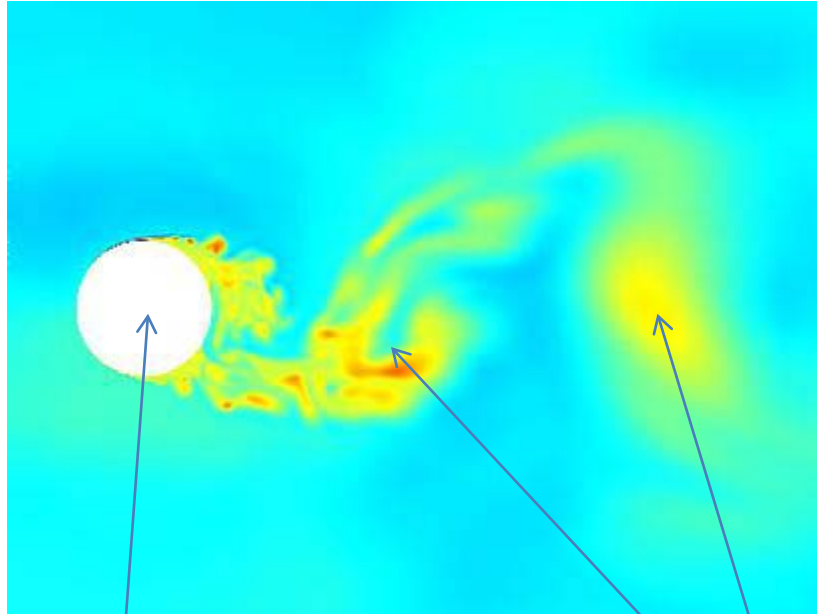


Gráficos por Computador. Un paso más adelante para generar representaciones de excepcional calidad, pero que no aporta calidad científica o interpretativa adicional.



Un exceso de énfasis en el “realismo” puede desviar la atención y el esfuerzo en la interpretación de los datos.

# Simulación de Flujo turbulento



Esfera

Vórtices, torbellinos

CFD: Computacional Fluid Dynamics

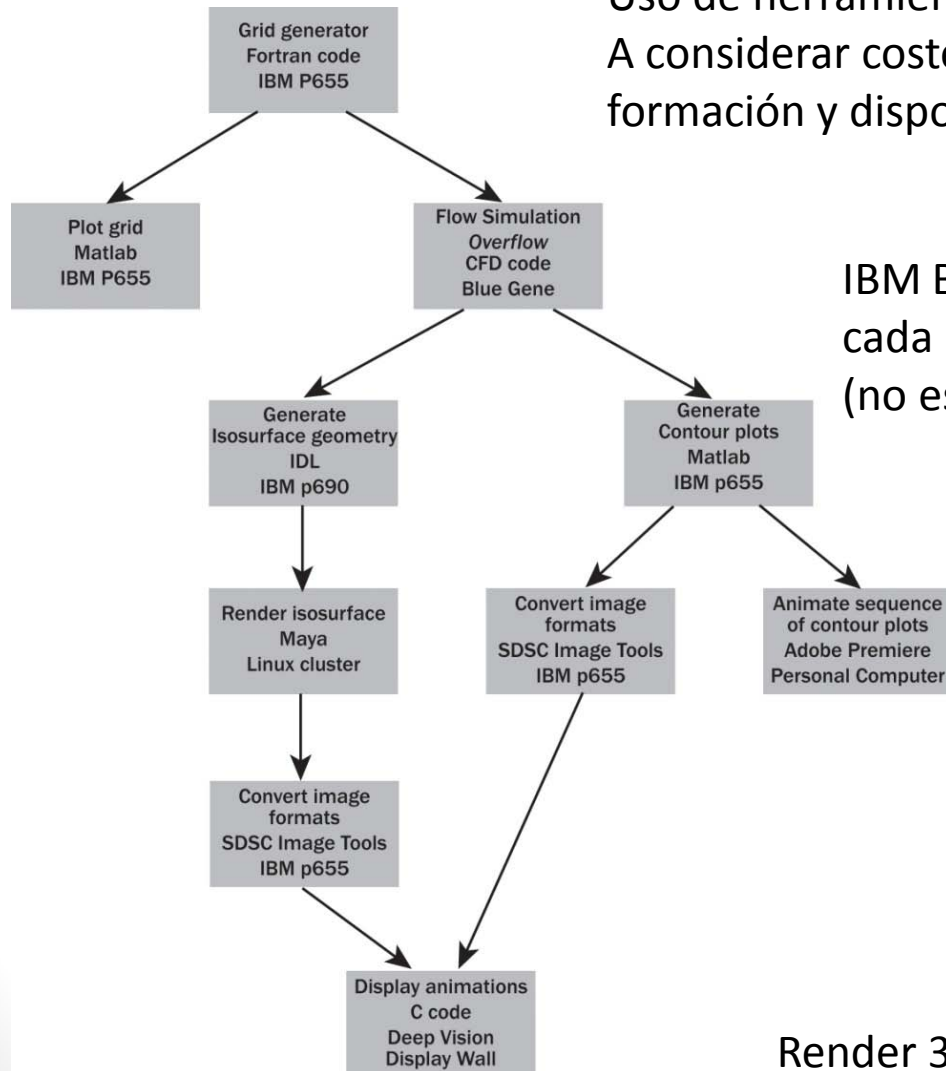
Simulación Computacional del flujo turbulento alrededor de una esfera que se desplaza a una cierta velocidad en el seno de un fluido (aire o agua) con régimen turbulento

Un mallado de  $401 \times 101 \times 301$  con un total de  $12e6$  puntos

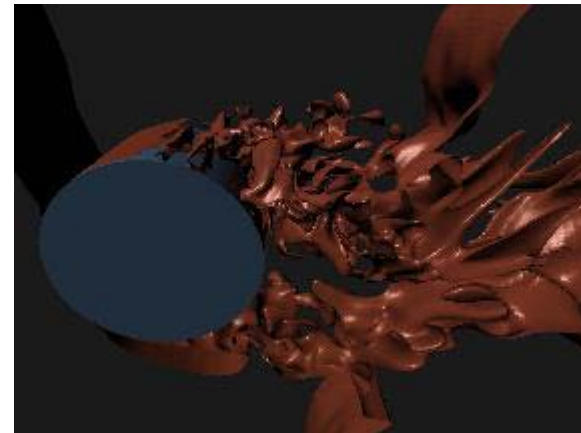


# Procesos en Simulación

Uso de herramientas heterogéneas.  
A considerar costes de adquisición, de  
formación y disponibilidad.



IBM Blue Gene, clúster con 256 nodos,  
cada nodo con 2 procesadores y 512M  
(no es lo más moderno)



Render 3D de superficies de iso-presión

# Herramientas de Visualización

Tool	Produce Input Data	Analyze, Filter, Reformat	Apply Sci Vis Techniques	Map to Geometry	Render	Postprocess	View Results
Experiments, Simulations	Y						
Custom code	x	x	x	x	x	x	x
MATLAB	x	Y	x	x	x		x
IDL	x	Y	x	x	x		x
VTK		x	Y	x	x		x
Paraview		x	Y	x	x		x
OpenGL					Y		x
Open Scene Graph					Y		x
Maya					Y		x
Photoshop						Y	x
Gimp						Y	x
Imagemagick						Y	x
Premier						Y	x
Journals, web browsers, Projectors							Y

Diversas herramientas, no todas cubren todos los aspectos que pudieran ser necesarios. MATLAB una opción equilibrada, no siempre la mejor

# Algunas herramientas

IDL: Interface Definition Language es un lenguaje para computación numérica y visualización, cuyo uso no está demasiado extendido (entornos de astronomía y física). Una colección de ejemplos de visualización de datos:

<http://www.idlcoyote.com/gallery/index.html>

Python: Es un lenguaje de tipo script al estilo de MATLAB que es una opción open-source muy valiosa para computo y visualización de datos.

R: Está más bien orientado a procesos y visualización de datos en estadística y minería de datos.

VTK: Visualización Toolkit, una opción muy útil para ser utilizada desde C, Java, Python para visualización de datos. Es OpenSource.

<http://www.vtk.org/>

OpenGL: El estándar 2D y 3D para gráficos, diseño y visualización. Más laborioso pues requiere una programación en detalle.

# Ploteado simple de datos

- Para la presentación visual sencilla de datos se pueden multiples herramientas sin demasiada sofisticación:
- MATLAB
- GNUPlot
- Excel
- Grace (Unix/Linux + X)

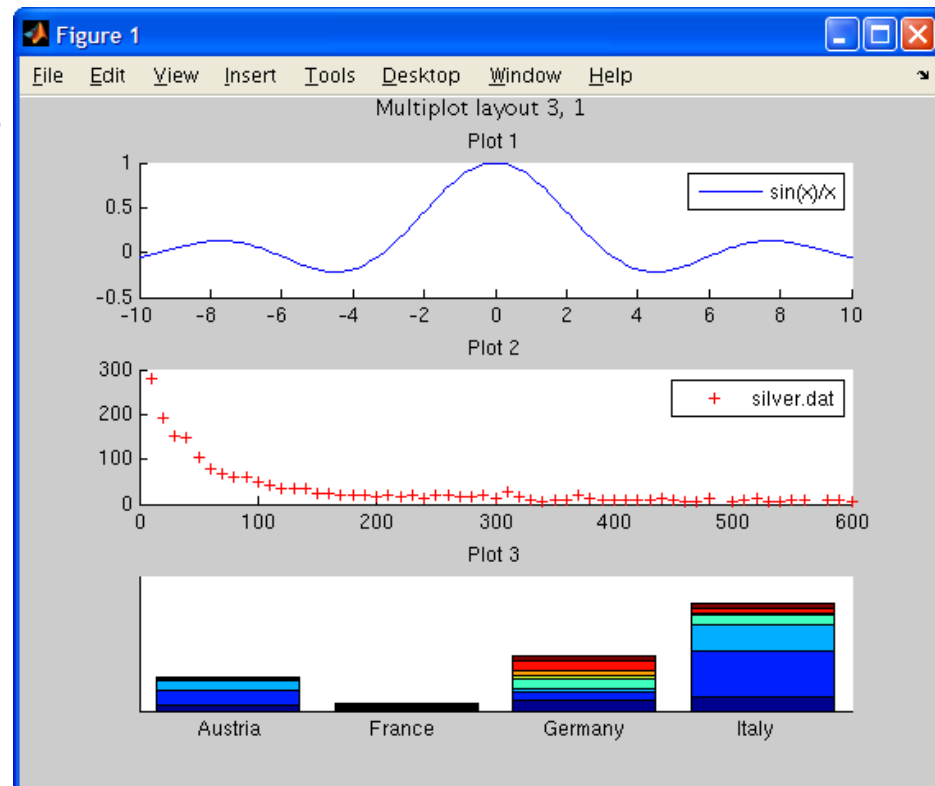
# Matlab

Un entorno orientado a la computación matricial que incorpora multitud de herramientas para la visualización científica.

Su principal ventaja es la sencillez de uso y su gran difusión.

Su principal inconveniente es que no es abierto, pero existe una versión OpenSource: Octave

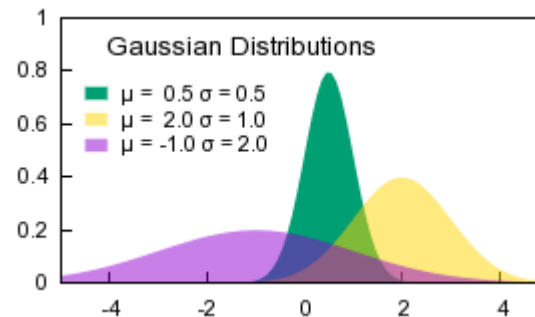
Consultar una introducción a MATLAB disponible en el Campus Virtual



# GNUPlot

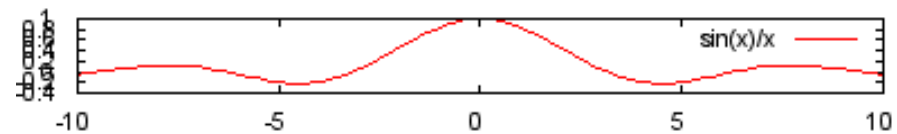
Un conjunto de primitivas que se ejecutan en forma de líneas de comando. Útil para plotado simple de datos:

<http://www.gnuplot.info/>

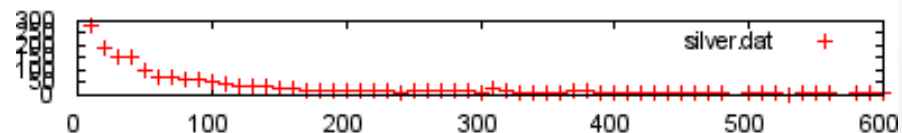


Multiplot layout 3,1

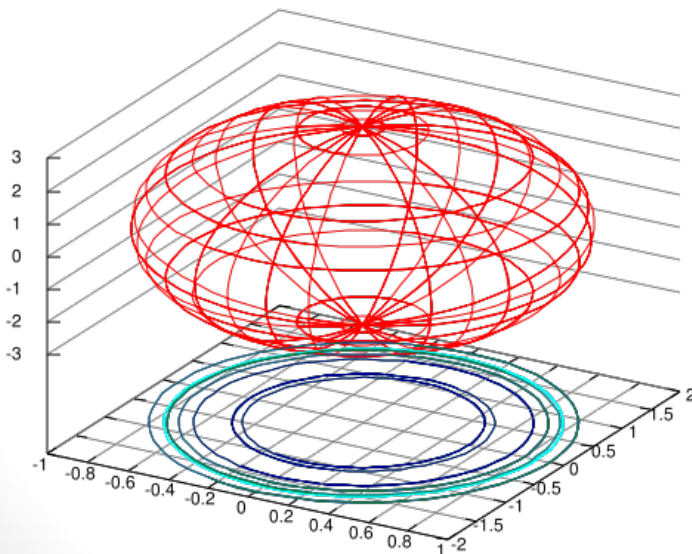
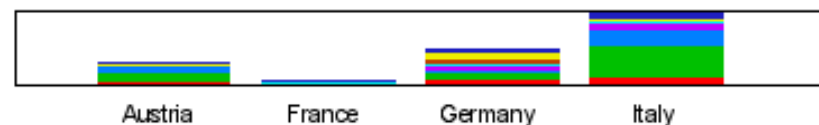
Plot 1



Plot 2

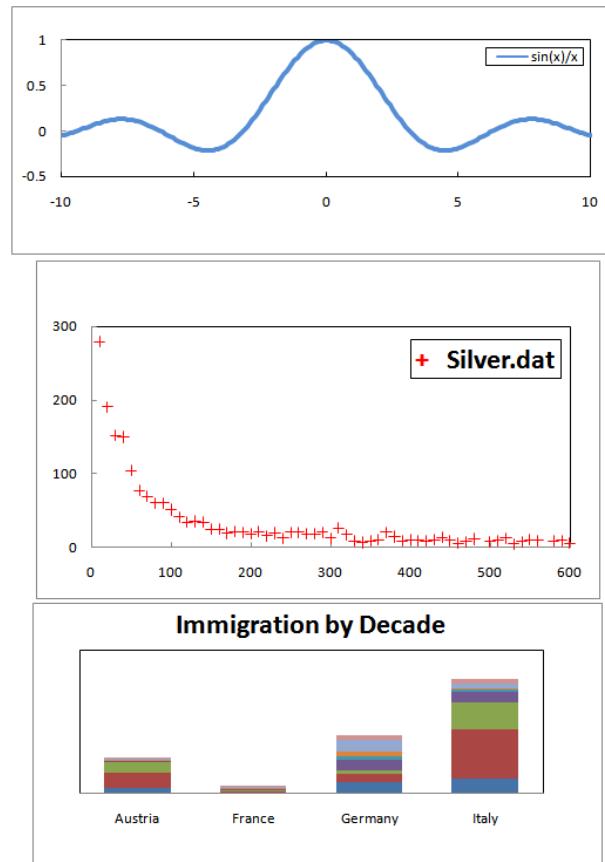


Plot 3



# Excel

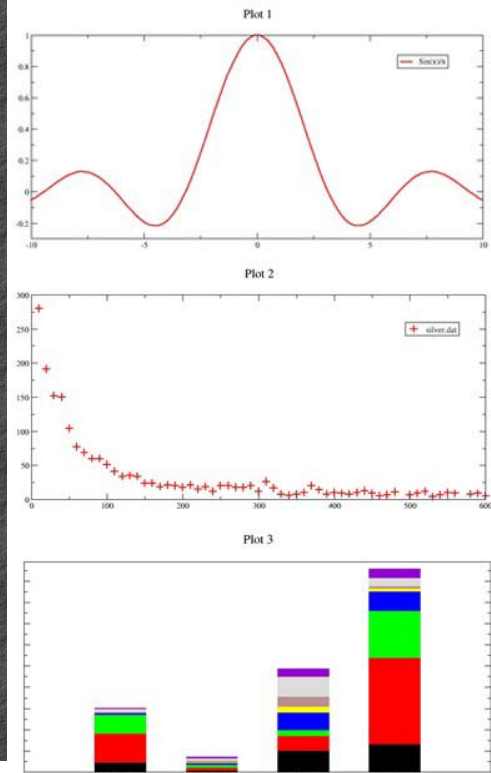
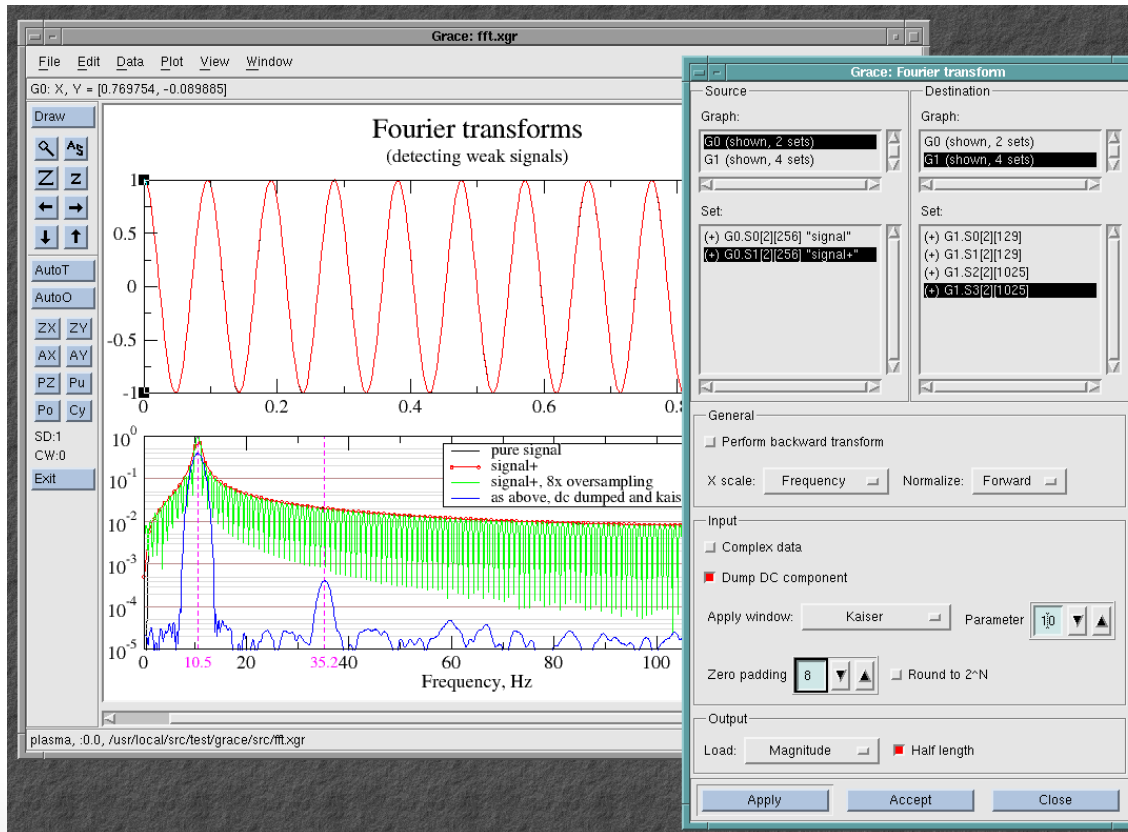
Ampliamente difundido y sencillo de utilizar para presentación visual de datos de naturaleza simple. Su ventaja sencillez y alta disponibilidad, su inconveniente ausencia de mecanismos avanzados de presentación visual.



# XMGrace

Entorno especialmente diseñado para X Windows, Unix/Linux. No muy difundido

<http://plasma-gate.weizmann.ac.il/Grace/>





# Visualización científica en MATLAB

- Permite una amplia variedad de tareas de visualización de datos, desde los más simples hasta los bastante sofisticados.
- Ploteado simple 2D y 3D
- Ploteado de superficies alabeadas.
- Visualización de imágenes.
- Visualización de resultados. Curvas de nivel, contornos.
- Visualización avanzada, campos y líneas de corrientes.
- Visualización de objetos, ...
- Para información avanzada consultar las referencias incluidas en este Curso en el Campus Virtual.

# Pautas de ploteado simple en MATLAB

- Creación de una ventana para la visualización, `figure()`;
- Orden de acumulación de resultados, `hold on/off`.
- Opcionalmente selección de subfiguras, `subplot`.
- Ploteado simple 2D o 3D, `plot()` o `plot3()`.
- Anotación de los ejes, `xlabel()` y `ylabel()`.
- Mallado de referencia, `grid on`
- Límites de los ejes, `axis()`
- Etiquetas de información de ploteados, `legend()`.
- En 3D, navegación tridimensional, `cameratoolbar`.
- Edición de postproducción.

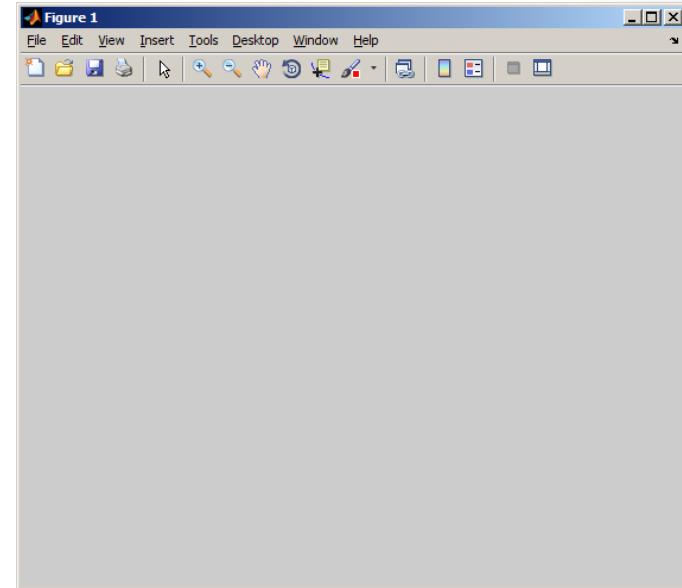
# figure

- Es el comando básico para crear una ventana que contendrá un gráfico para representar datos. Las distintas figuras se seleccionan por un número.
- `figure`: crea una ventana y le asigna un número
- `figure(1)` permite seleccionar una figura como la actual
- `h=gcf` permite obtener el handle `h` de una figura (es un número entero)
- `get(h)` proporciona todas las propiedades
- `propiedad = get(h, 'nombre propiedad')`
- `set(h, 'nombre propiedad', valor de la propiedad);`
- Muchas propiedades pueden ser modificadas con comando específicos.

# Propiedades de figure:

## Command Window

```
>> figure;  
>> h = gcf;  
>> get(h)  
    Alphamap = [ (1 by 64) double array]  
    CloseRequestFcn = closereq  
    Color = [0.8 0.8 0.8]  
    Colormap = [ (64 by 3) double array]  
    CurrentAxes = []  
    CurrentCharacter =  
    CurrentObject = []  
    CurrentPoint = [0 0]  
    DockControls = on  
    FileName =  
    FixedColors = [ (3 by 3) double array]  
    IntegerHandle = on  
    InvertHardcopy = on  
    KeyPressFcn =  
    KeyReleaseFcn =  
    MenuBar = figure  
    MinColormap = [64]  
    Name =  
    NextPlot = add  
    NumberTitle = on  
    PaperUnits = centimeters
```



Subplot: permite fraccionar la figura en un array de zonas parciales donde plotear datos .

subplot( m,n,p) selecciona como zona activa el elemento p en una array de subfiguras de m filas y n columnas

# Creación del espacio de la variable

- $A = \text{inicial}:\text{step}:\text{final};$  creación de espacio de valores
- $y=f(x)$ , funciones explícitas.
- $x = \text{linspace}(x_{\text{inicial}}, x_{\text{final}}, \text{numero de puntos})$
- $y = x.^2$ , usar operaciones vectoriales: con el prefijo  $.$
- $x = x(t)$ , funciones paramétricas
- $y = y(t)$
- $t = \text{linspace}(t_{\text{inicial}}, t_{\text{final}}, \text{numero de puntos})$
- $x = t;$
- $y = t.^2;$  en ambos casos utilizar operaciones vectoriales.

# Esquema simple 2D

figure;

hold on;      hold on / hold off permite la agregación de diversos plotados a una misma figura

% diversos plot()

xlabel('anotación del eje x');      Define el nombre de la variable de los ejes

ylabel('anotación del eje y');

grid on;      Permite ver el mallado de referencia de los ejes

% axis equal      Axis permite cambiar propiedad de los ejes, por  
axis([xmin,xmax,ymin,ymax]);      ejemplo definir los límites de visualización

legend('texto1','texto2',...);      Agregación de textos explicativos de cada gráfico

title('titulo de la figura')

hold off;

# plot

- `plot(x,y,marcas)`
- Marcas = 'color + marca + unión'
- 'bo-' color azul + marcas círculos + líneas continuas

Specifier	Marcas	Marker
o		Circle
+		Plus sign
*		Asterisk
.		Point
x		Cross
s		Square
d		Diamond
^		Upward-pointing triangle
v		Downward-pointing triangle
>		Right-pointing triangle
<		Left-pointing triangle
p		Pentagram
h		Hexagram

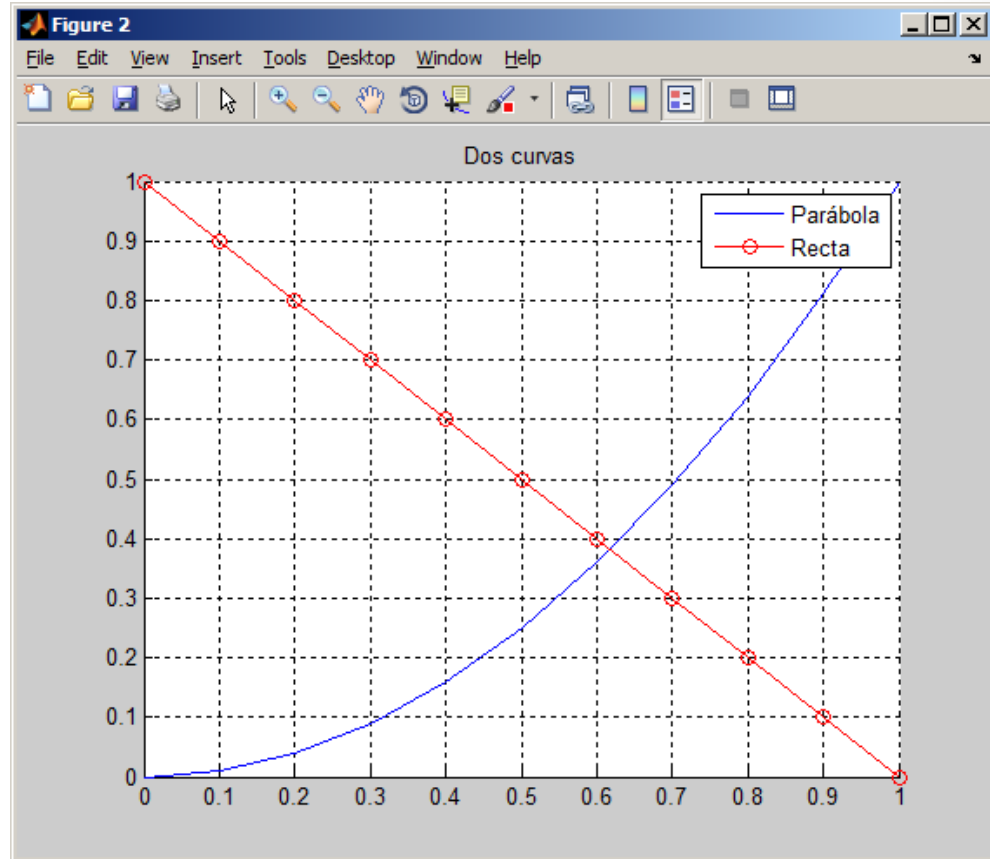
Specifier	Unión	Line Style
-		Solid line (default)
--		Dashed line
:		Dotted line
-.		Dash-dot line

Specifier	Color
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

# Ejemplo simple 2D

## Command Window

```
>> figure;  
>> hold on;  
>> x = 0:0.1:1;  
>> y = x.^2;  
>> plot(x,y,'b-');  
>> plot(x,1-x,'ro-');  
>> grid on;  
>> legend('Parábola','Recta');  
>> title('Dos curvas');  
>> hold off;  
>>
```





# Esquema simple 3D

```
figure;  
hold on;
```

```
% diversos plot3()
```

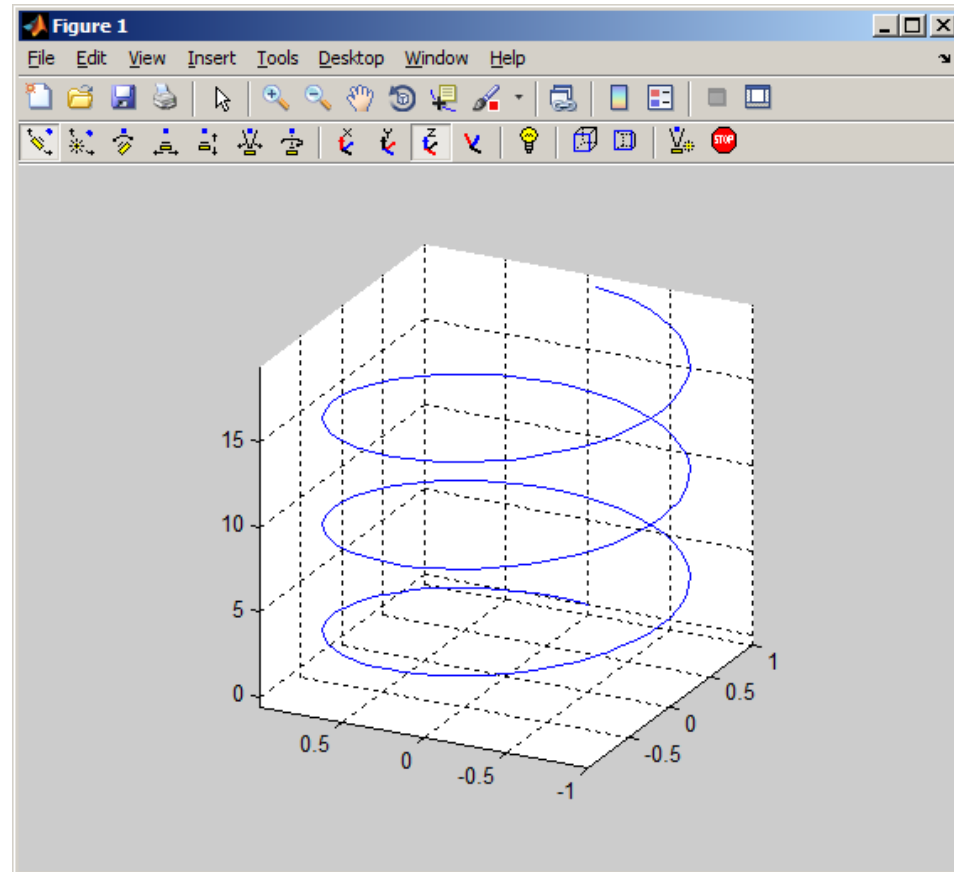
```
xlabel('anotación del eje x');  
ylabel('anotación del eje y');  
zlabel('anotación del eje z');  
axis([xmin,xmax,ymin,ymax]);  
%axis equal;  
hold off;
```

```
cameratoolbar;    Herramienta que permite “navegar” tridimensionalmente  
                  alrededor de la figura
```

# Ejemplo simple 3D. Curva paramétrica

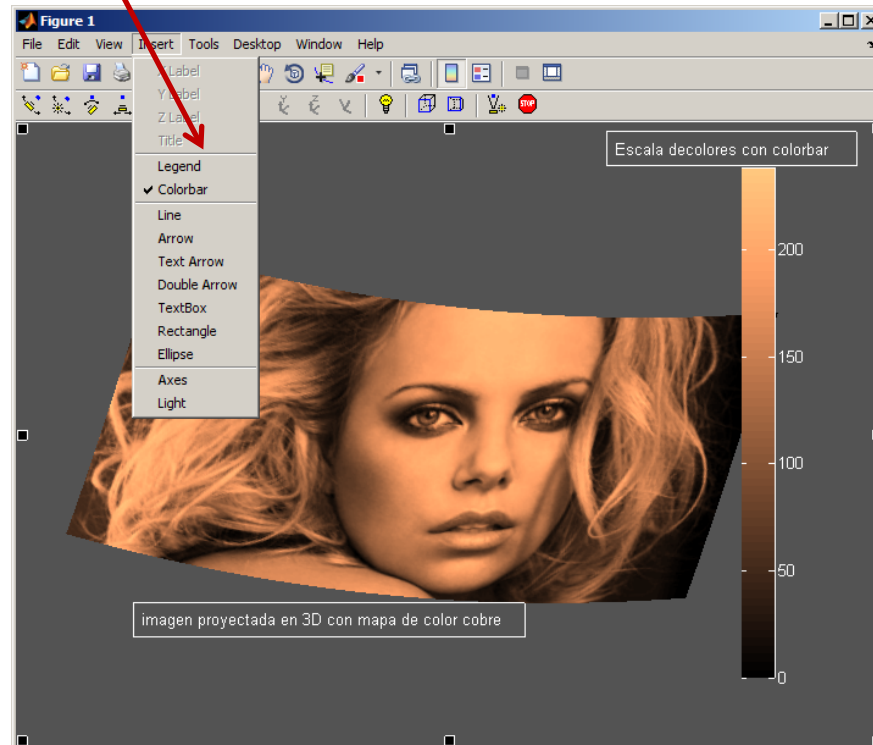
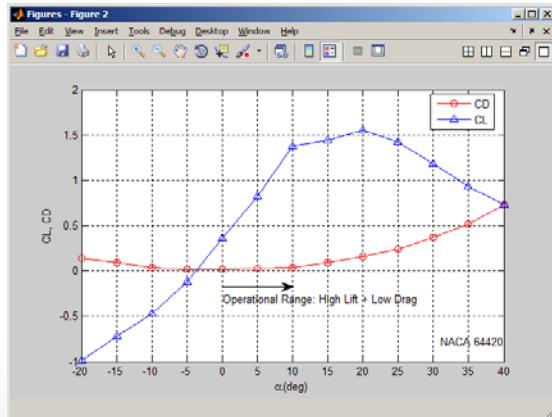
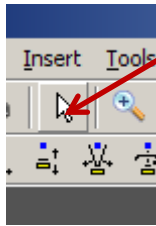
## Command Window

```
>> figure;  
>> hold off;  
>> t = 0:0.1:6*pi;  
>> x = cos(t);  
>> y = sin(t);  
>> z = t;  
>> plot3(x,y,z,'b-');  
>> grid on;  
>> cameratoolbar;  
>> hold off;  
fx >>
```



# Edición y Postprocesado

- Inclusión de elementos adicionales de visualización (líneas, flechas, textos, ...)
- Modificación de las propiedades de los objetos gráficos.



# Espacio lineal 2d

- Útil para la generación de superficies alabeadas, dependientes de dos variables  $(u,v)$ , en la forma:  $x=x(u,v)$ ,  $y=y(u,v)$ ,  $z=z(u,v)$ .
- 1. Generación de un espacio lineal de las variables:
- $[U,V]=\text{meshgrid}(\text{umin}:\text{ustep}:\text{umax},\text{vmin}:\text{vstep}:\text{vmax})$
- 2. cálculo de las funciones  $X,Y,Z$  en función de  $U,V$
- 3. Visualización:  $\text{surf}(X,Y,Z)$

$$X = f(U,V)$$

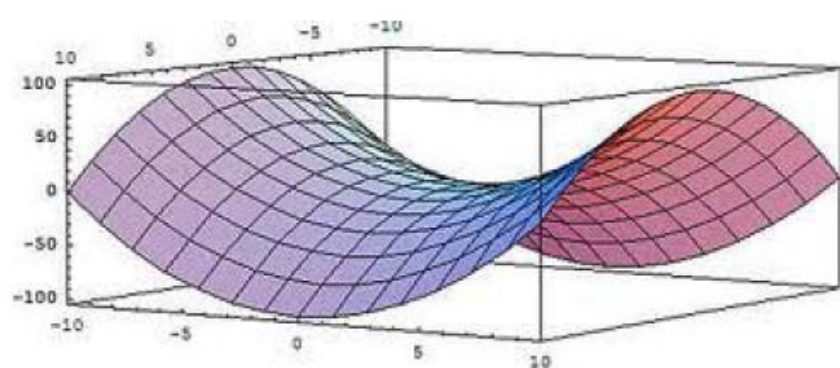
$$Y = g(U,V)$$

$$Z = h(U,V)$$

```
u=linspace(umin,umax,Nu);
```

```
v=linspace(vmin,vmax,Nv);
```

```
[U,V]= meshgrid(u,v);
```





# Visualización de un toro

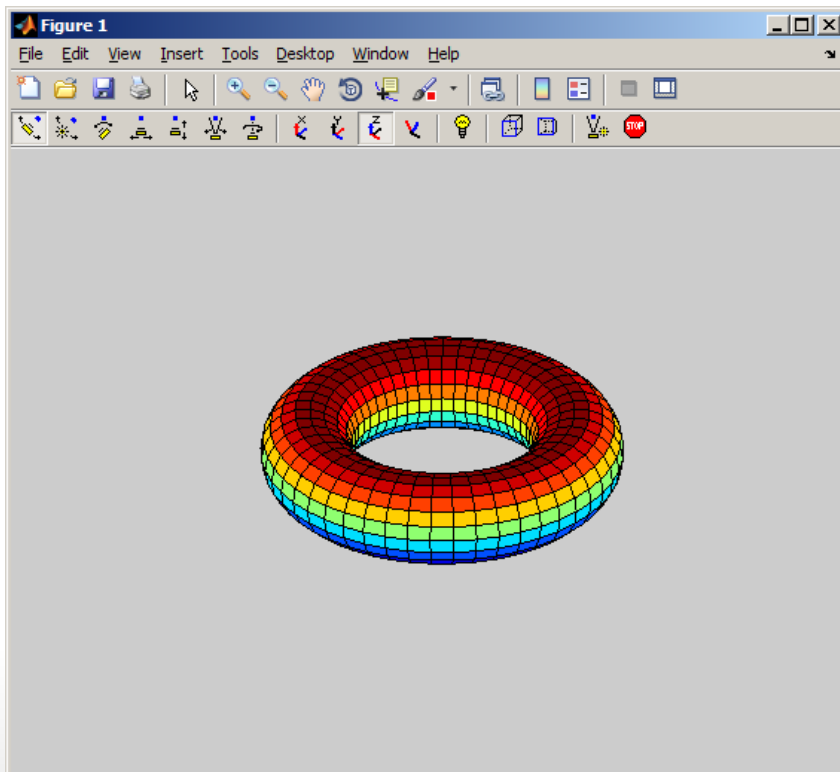
El programa es similar pero en este caso las ecuaciones que describen los puntos de la superficie están definidas como sigue en base a dos radios,  $R_1 > R_2$

$$x = (R_1 + R_2 \cos \theta) \cos \varphi$$

$$y = (R_1 + R_2 \cos \theta) \sin \varphi$$

$$z = R_2 \sin \theta$$

Donde los parámetros son:  $\theta \in [0, 2\pi]$  y  $\varphi \in [0, 2\pi]$ . El programa M



```
Editor - C:\Users\Juan\Documents\MATLAB\toro.m
File Edit Text Go Cell Tools Debug Desktop >> >> X
+ + - 1.0 + ÷ 1.1 x % + % - i
1 - clear all;
2 - clc;
3 - R1 = 3;
4 - R2 = 1;
5 - Nt = 20;
6 - Np = 50;
7 - theta = linspace(0,2*pi,Nt);
8 - phi = linspace(0,2*pi,Np);
9 - [Theta,Phi]= meshgrid(theta,phi);
10 - X = (R1+R2.*cos(Theta)).*cos(Phi);
11 - Y = (R1+R2.*cos(Theta)).*sin(Phi);
12 - Z = R2.*sin(Theta);
13 - surf(X,Y,Z);
14 - axis equal;
15 - axis off;
16 - cameratoolbar;
17
18
script Ln 15 Col 10 OVR
```

# Escalas de colores: colormap y shading

Explicar en la pizarra detalladamente el sistema de colores (pseudo-color) de MATLAB en la función surf

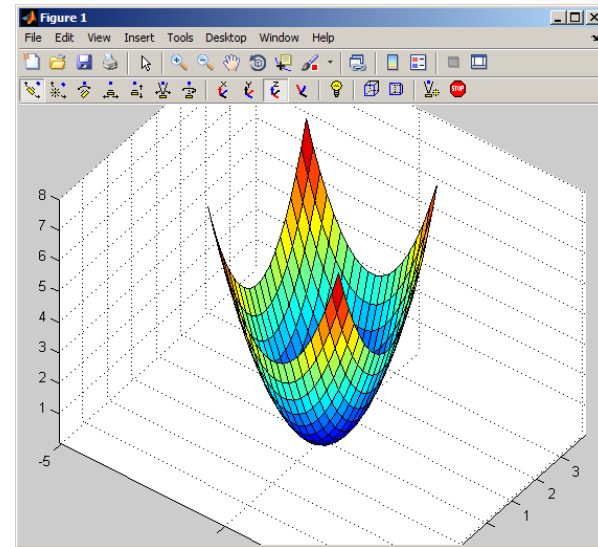
- Colormap gray, hot, cool, bone, copper, pink, flag, prism, jet
- Surf(x,y,z,c); para definir colores específicos.
- colormap(tabla); para definir una tabla de color
- tabla = colormap; para obtener la tabla actual
- Colorbar; para visualizar la tabla de colores.
- Shading flat color constante en cada faceta
- Shading interp color interpolado en los pixeles
- Shading faceted es shading flat superpuesto con las líneas de las facetas, es el valor de defecto
- Para más información: help graph3d

# Ejemplo de diversos mapas de colores

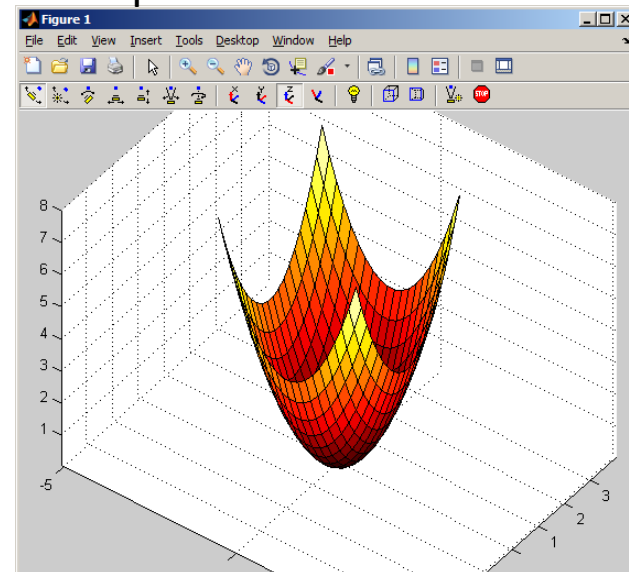
Colormap jet por defecto, la más recomendada para visualización

```
Command Window

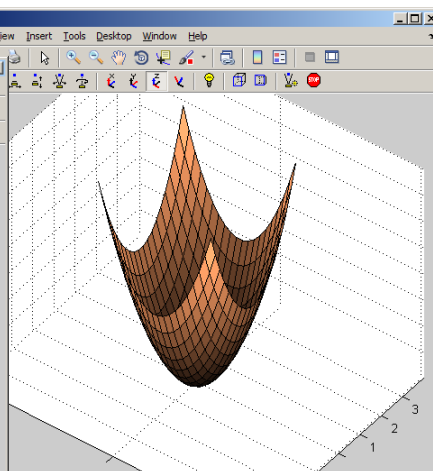
>> figure;
>> hold on;
>> x = linspace(-2,2,20);
>> y = linspace(-2,2,20);
>> [X,Y] = meshgrid(x,y);
>> Z = X.^2 + Y.^2;
>> surf(X,Y,Z);
>> cameratoolbar;
>> axis equal;
>> grid on;
>> |
```



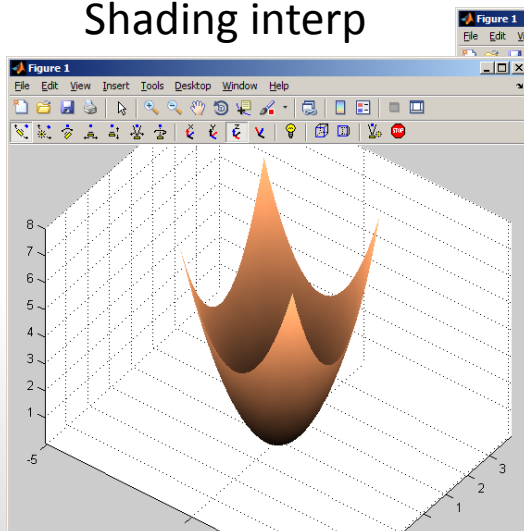
Colormap hot



Colormap copper



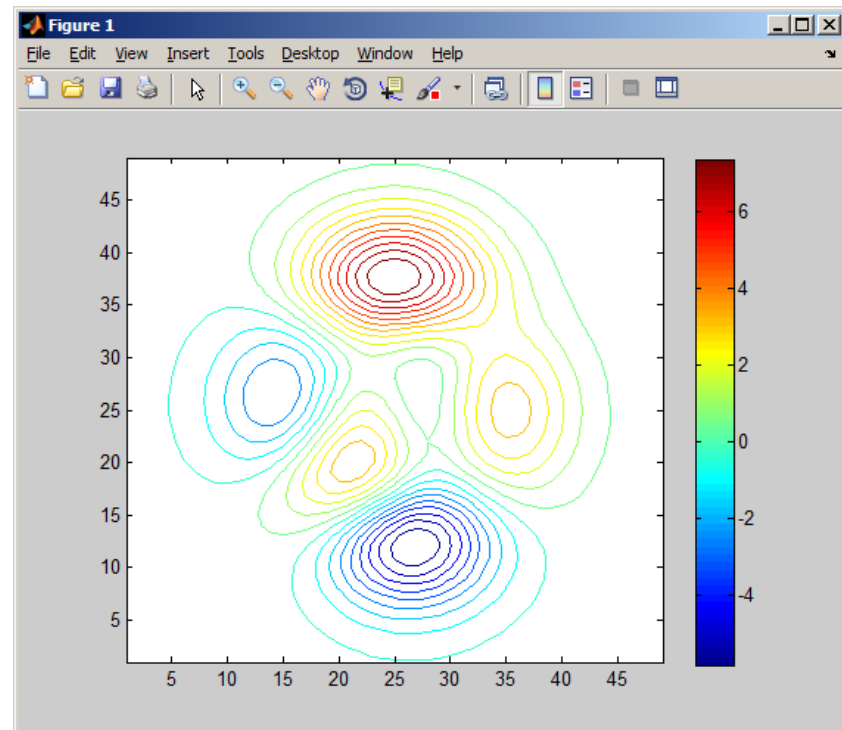
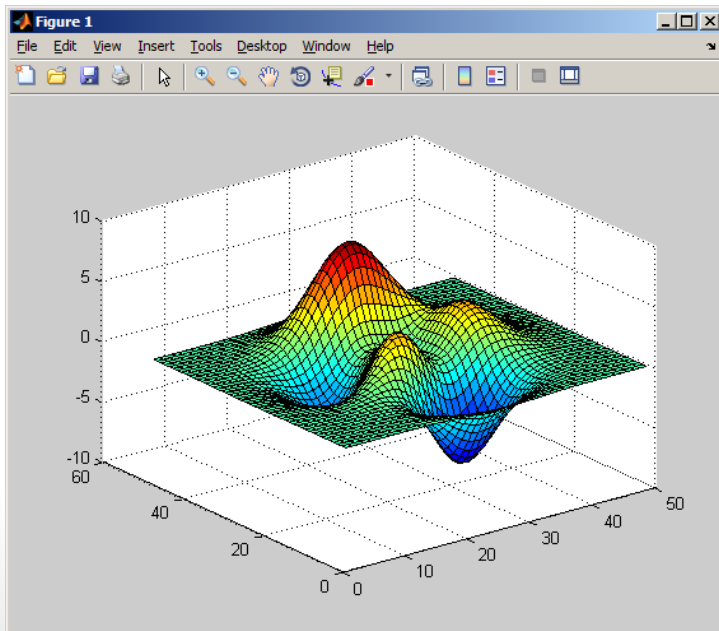
Shading interp





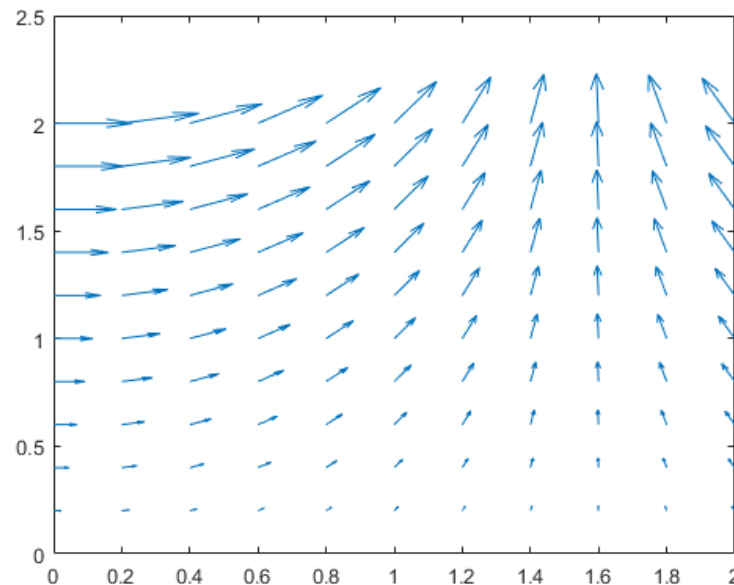
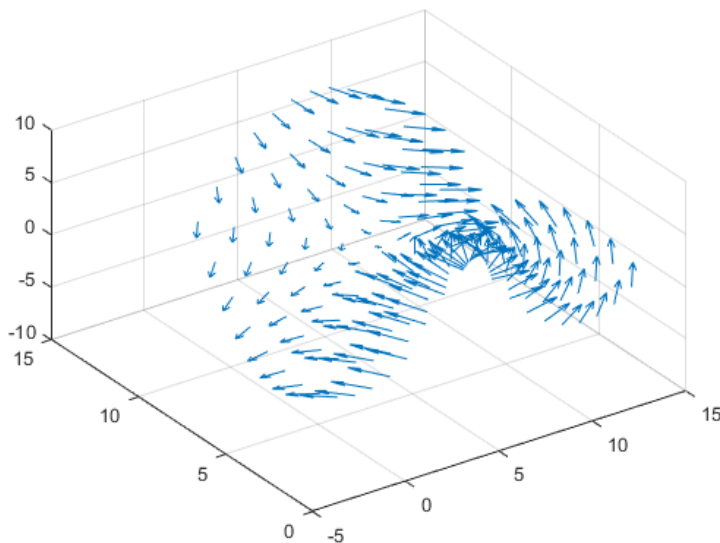
# Mapas de contornos.

- Contour
- `[c,h] = contour(peaks); clabel(c,h), colorbar`
- `[x,y,z]=peaks;` Peaks es una función de MATLAB de referencia
- `Surf(x,y,z);`
- `Contour(x,y,z,n);colorbar;`



# Campos vectoriales 2D y 3D

- $\text{Quiver}(x,y,u,v)$
- $\text{Quiver3}(x,y,z,u,v,w)$
- Conjuntos de puntos y valores de un vector en cada punto = campo vectorial

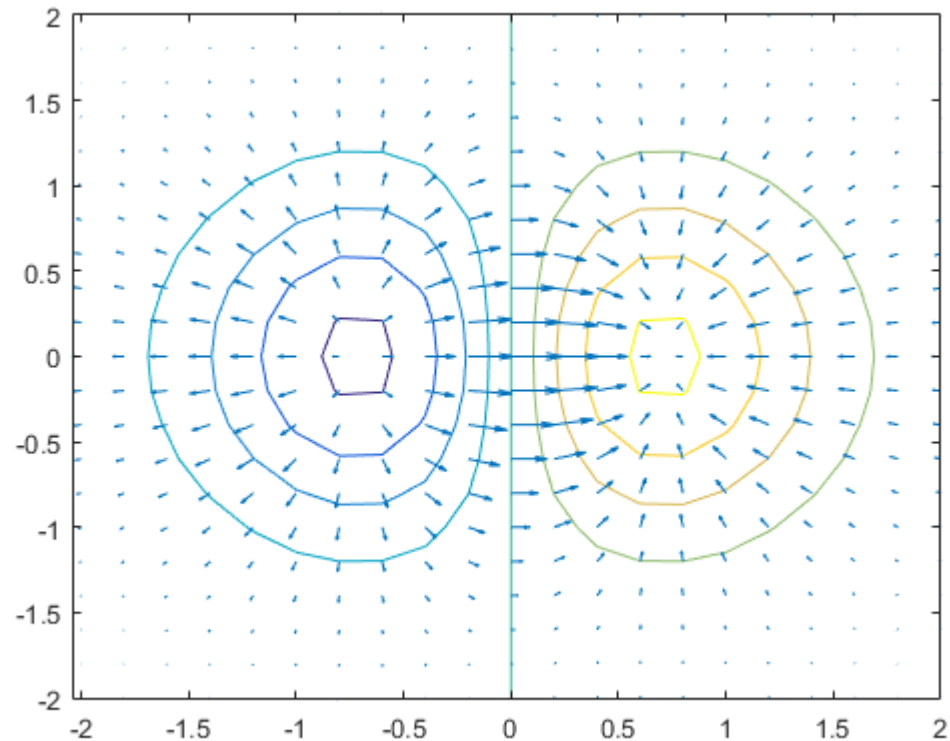


# Contornos (campo escalar)+gradiente(campo vectorial)

Plot the gradient of the function  $z = xe^{-x^2-y^2}$ .

```
[X,Y] = meshgrid(-2:.2:2);  
Z = X.*exp(-X.^2 - Y.^2);  
[DX,DY] = gradient(Z,.2,.2);
```

```
figure  
contour(X,Y,Z)  
hold on  
quiver(X,Y,DX,DY)  
hold off
```



# Líneas de corrientes vectoriales

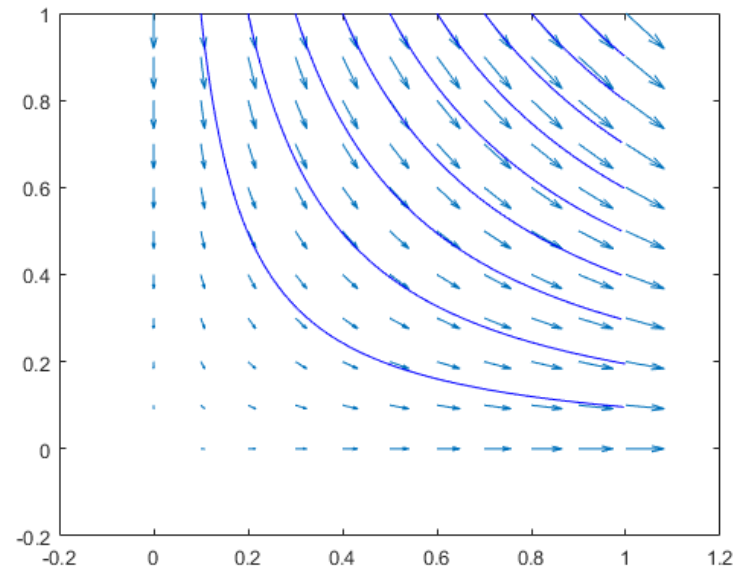
- `Streamline(x,y,u,v,startx,starty)`
- `Streamline(x,y,z,u,v,w,startx,starty,startz)`
- Puntos+valor del campo vectorial+conjuntos de comienzos de las líneas de corrientes.

Define arrays x, y, u, and v.

```
[x,y] = meshgrid(0:0.1:1,0:0.1:1);  
u = x;  
v = -y;
```

Create a quiver plot of the data. Plot streamlines that start at different points along the line  $y = 1$ .

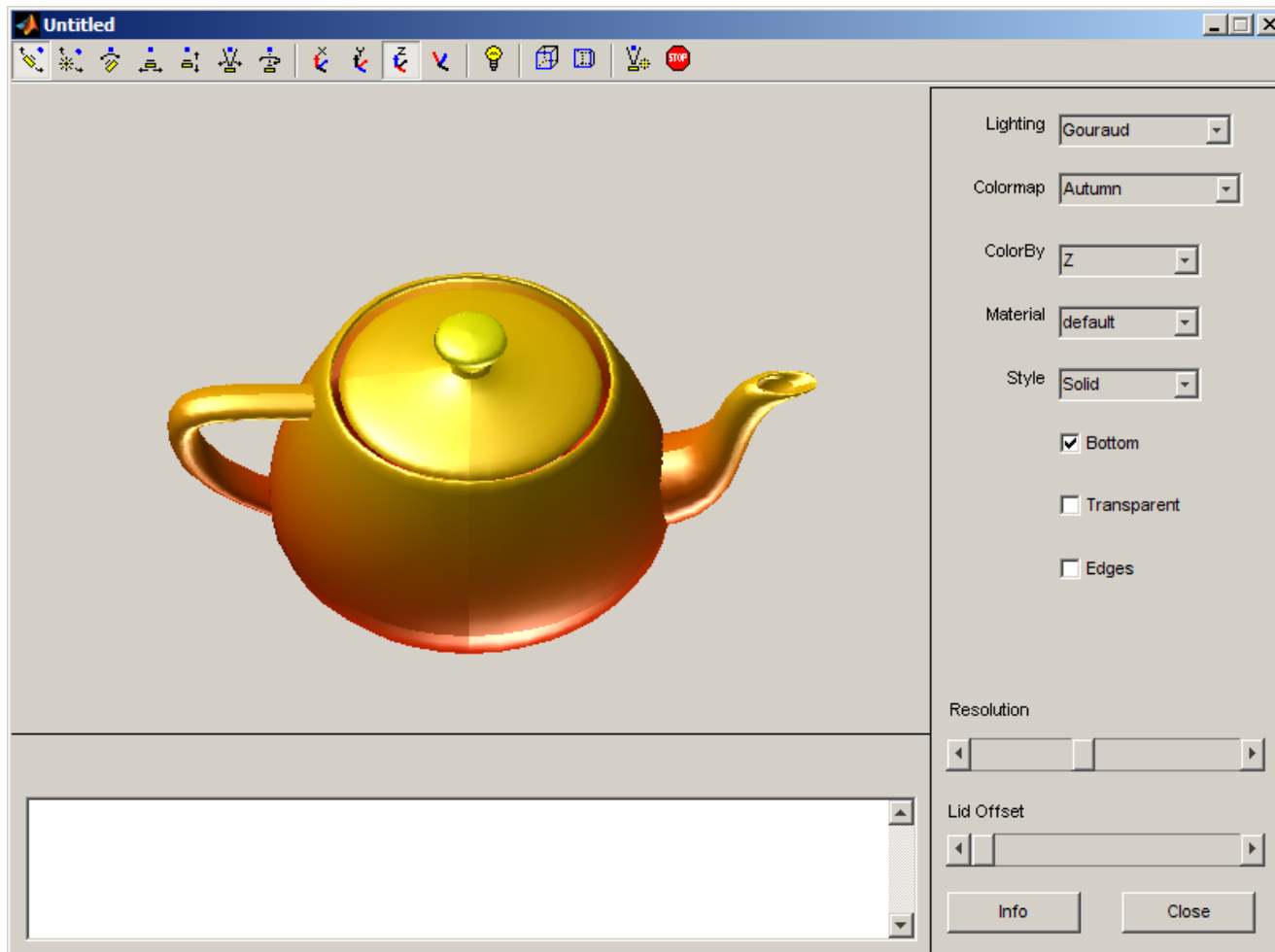
```
figure  
quiver(x,y,u,v)  
  
startx = 0.1:0.1:1;  
starty = ones(size(startx));  
streamline(x,y,u,v,startx,starty)
```



# Materiales e iluminación

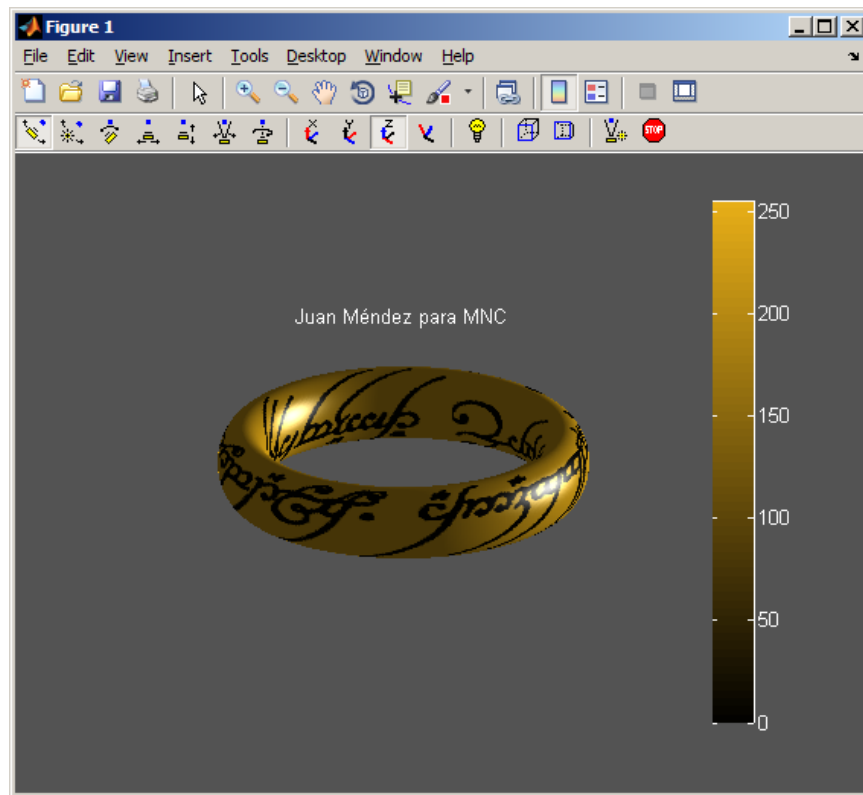
- Material shiny (brillante), dull(mate), metal modifica las características de reflectancia de la luz de la superficie, produciendo la apariencia de diferentes materiales.
- Lightening flat, gourand, phog diversos modelos de calculo de la reflexión de la luz en la superficie.

# Ejemplo: teapotdemo



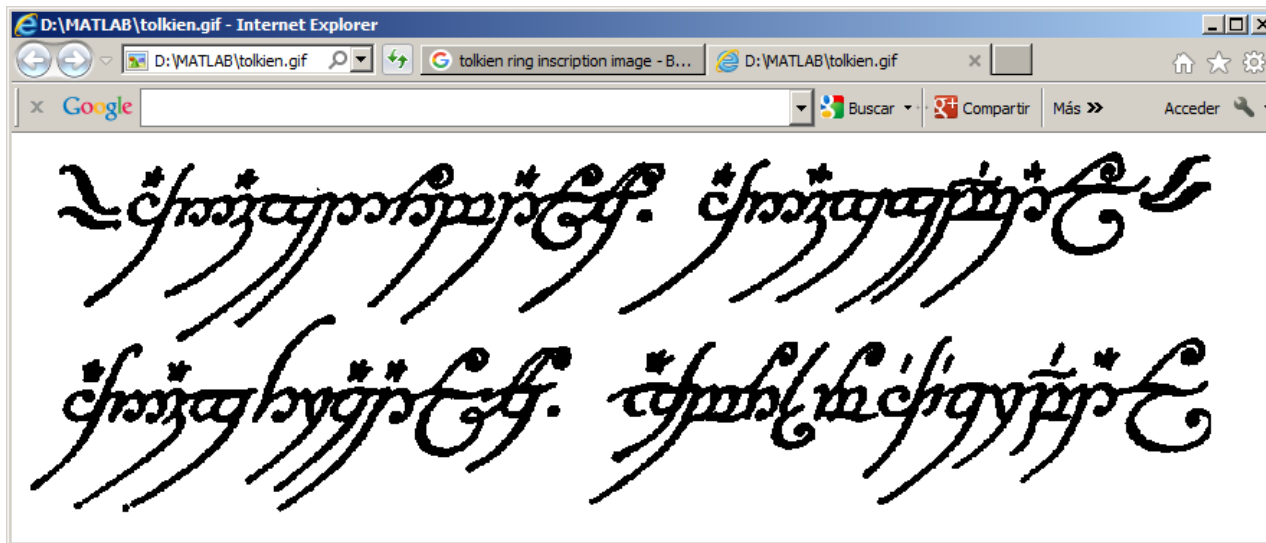
# Ejemplo: Casi el anillo de Sauron en MATLAB.

- Ejemplo de ilustración de las posibilidades de visualización utilizando sencillos programas MATLAB.
- Ejemplo divertimento:



# One Ring: Paso 1

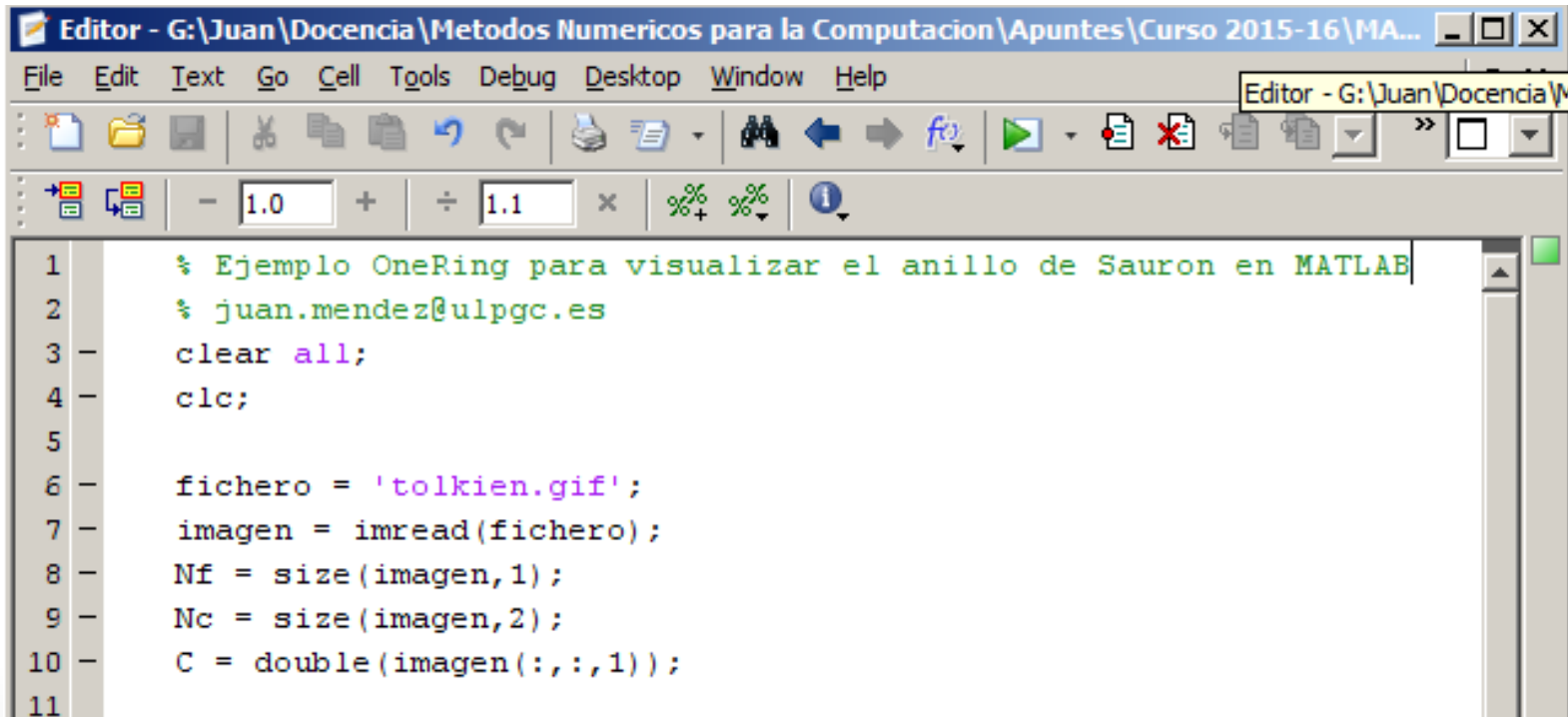
- Obtener una imagen del texto de la inscripción del anillo, por ejemplo buscando en Google: **tolkien ring inscription image** en blanco y negro y poca resolución. El fichero tolkien.gif que se adjunta lo incorpora.





# One Ring: Paso 2

- Leer la imagen y descubrir sus dimensiones para ser mapeada en un toro con la forma del anillo



The image shows a MATLAB Editor window with the following code:

```
1 % Ejemplo OneRing para visualizar el anillo de Sauron en MATLAB
2 % juan.mendez@ulpgc.es
3 clear all;
4 clc;
5
6 fichero = 'tolkien.gif';
7 imagen = imread(fichero);
8 Nf = size(imagen,1);
9 Nc = size(imagen,2);
10 C = double(imagen(:,:,1));
11
```

# OneRing: Paso 3

- Definir la geometría de un toro con sección elíptica sobre la que proyectar la imagen. La geometría del toro se debe acomodar a las dimensiones de la imagen

```
12 - R2 = 1;  
13 - R1 = 2*Nc*R2/Nf;  
14 - theta = linspace(0,2*pi,Nf);  
15 - phi = linspace(0,2*pi,Nc);  
16 - [Theta,Phi]= meshgrid(theta,phi);  
17 - X = (R1+R2.*sin(Theta)).*cos(Phi);  
18 - Y = (R1+R2.*sin(Theta)).*sin(Phi);  
19 - Z = 1.5*R2.*cos(Theta);
```



Geometría elíptica, probar con valores diferentes: 1.7, 2.0, ...

# OneRing: Paso 4

- Definir el mapa de colores, el tipo de material y proyectar la imagen en la superficie de la geometría.
- El color oro pepita se define en la actividad práctica.

```
21 %colormap copper;  
22 - colormap(oro pepita);  
23 - material metal;  
24 - whitebg('black');  
25  
26 - surf(X,Y,Z,C');  
27 - shading interp;  
28 - axis equal;  
29 - axis off;  
30 - colorbar;  
31 - cameratoolbar;  
32 - title('Juan Méndez para MNC')  
33
```

# OneRing: Paso 5



- Pero la inscripción debería ser más clara que la superficie del anillo.
- Solución: cambiar los valores de la imagen de la inscripción y/o de la tabla de colores. Sugerencia: colormapeditor
- Lo realizará el alumno como una actividad de la Práctica 1. Pero evidentemente al nivel de MATLAB no se puede alcanzar prestaciones de gran realismo, no es su función.
- Recordar: **La visualización Científica no trata de alcanzar los niveles de realismo que si se pueden requerir en aplicaciones de Computer Graphics avanzadas.**

# Bibliografía

Boston University. Information Services and Technologies. Tutorial.

<http://www.bu.edu/tech/support/research/training-consulting/online-tutorials/>

Introduction to Scientific Visualization Tutorial.

<http://www.bu.edu/tech/support/research/training-consulting/online-tutorials/introduction-to-scientific-visualization-tutorial/>

Using MATLAB to Visualize Scientific Data (online tutorial)

<http://www.bu.edu/tech/support/research/training-consulting/online-tutorials/visualization-with-matlab/>

MATLAB Graphics and Data Visualization Cookbook, N. Majundar, S. Banerjee,

PACKT Pub, 2012. [http://math-cs.aut.ac.ir/~shamsi/Matlab/matlab\\_graphics\\_and\\_data\\_visualization\\_cookbook.pdf](http://math-cs.aut.ac.ir/~shamsi/Matlab/matlab_graphics_and_data_visualization_cookbook.pdf)